

cket No.: 57454-060

*#2
Priority
Page
MMA
6/4/01*
PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of

Isao MINEMATSU

Serial No.:

Group Art Unit:

Filed: March 29, 2001

Examiner:

For: MICROPROCESSOR EXECUTING DATA TRANSFER BETWEEN MEMORY AND REGISTER AND DATA TRANSFER BETWEEN REGISTERS IN RESPONSE TO SINGLE PUSH/POP INSTRUCTION



**CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Commissioner for Patents
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims the priority of:

Japanese Patent Application No. 2000-194033,
filed June 28, 2000

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

[Signature]
Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:dtb
Date: March 29, 2001
Facsimile: (202) 756-8087

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

57454-060

Mitsubishi

March 29, 2001

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application:

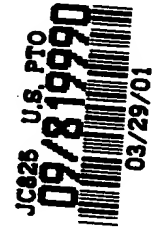
2000年 6月28日

出願番号
Application Number:

特願2000-194033

出願人
Applicant(s):

三菱電機株式会社

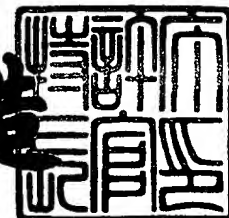


CERTIFIED COPY OF
PRIORITY DOCUMENT

2000年 7月21日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2000-3057733

【書類名】 特許願
【整理番号】 523187JP01
【提出日】 平成12年 6月28日
【あて先】 特許庁長官殿
【国際特許分類】 G06F 9/22
G06F 9/30
G06F 9/34
G06F 9/44

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社
社内

【氏名】 峯松 勲

【特許出願人】

【識別番号】 000006013

【氏名又は名称】 三菱電機株式会社

【代理人】

【識別番号】 100064746

【弁理士】

【氏名又は名称】 深見 久郎

【選任した代理人】

【識別番号】 100085132

【弁理士】

【氏名又は名称】 森田 俊雄

【選任した代理人】

【識別番号】 100091409

【弁理士】

【氏名又は名称】 伊藤 英彦

【選任した代理人】

【識別番号】 100096781

【弁理士】

【氏名又は名称】 堀井 豊

【選任した代理人】

【識別番号】 100096792

【弁理士】

【氏名又は名称】 森下 八郎

【手数料の表示】

【予納台帳番号】 008693

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 マイクロプロセッサ並びにアセンブラ、その方法およびそのプログラムを記録した記録媒体

【特許請求の範囲】

【請求項 1】 命令コードのフェッチを制御するための制御手段と、
前記フェッチされた命令コードをデコードするためのデコード手段と、
前記デコード手段によるデコード結果に基づいて、メモリのアドレスを演算するためのアドレス演算手段と、

前記デコード手段によるデコード結果に基づいて、データを演算するためのデータ演算手段とを含み、

前記データ演算手段は、前記制御手段によってフェッチされた 1 つのオペレーションコードを有する 1 つの命令コードに対応して、レジスタ間のデータ転送と、レジスタとメモリとの間のデータ転送とを実行する、マイクロプロセッサ。

【請求項 2】 前記データ演算手段は、前記制御手段によってフェッチされた 1 つの退避命令に対応して、第 1 のレジスタに格納されたデータをメモリに転送し、第 2 のレジスタに格納されたデータを前記第 1 のレジスタに転送する、請求項 1 記載のマイクロプロセッサ。

【請求項 3】 前記データ演算手段は、前記第 2 のレジスタに格納されたデータを前記第 1 のレジスタに転送した後、スタックポインタの値をデクリメントする、請求項 2 記載のマイクロプロセッサ。

【請求項 4】 前記データ演算手段は、前記制御手段によってフェッチされた 1 つの復帰命令に対応して、第 1 のレジスタに格納されたデータを第 2 のレジスタに転送し、メモリに格納されたデータを前記第 1 のレジスタに転送する、請求項 1 記載のマイクロプロセッサ。

【請求項 5】 前記データ演算手段は、前記メモリに格納されたデータを前記第 1 のレジスタに転送した後、スタックポインタの値をインクリメントする、請求項 4 記載のマイクロプロセッサ。

【請求項 6】 前記第 1 のレジスタは、前記データ演算手段に実装されるワークレジスタである、請求項 2 ～ 5 のいずれかに記載のマイクロプロセッサ。

【請求項 7】 前記第 2 のレジスタは、前記アドレス演算手段または前記制御手段に実装される制御レジスタである、請求項 2 ～ 6 のいずれかに記載のマイクロプロセッサ。

【請求項 8】 前記データ演算手段は、前記制御手段によってフェッチされた 1 つの退避命令に対して、第 1 のレジスタに格納されたデータをメモリに転送し、スタックポインタの値をそのまま保持するための手段を含む、請求項 1 ～ 7 のいずれかに記載のマイクロプロセッサ。

【請求項 9】 ソースプログラムからコードを読み込むためのコード読込手段と、

複数のレジスタを特定するための情報を記憶するための記憶手段と、

前記コード読込手段によって読み込まれたコードが第 1 のマクロ命令である場合、前記コードに含まれる複数のレジスタを特定するための情報を前記記憶手段に記憶し、前記複数のレジスタを退避するコードを生成するための第 1 のコード生成手段と、

前記コード読込手段によって読み込まれたコードが第 2 のマクロ命令である場合、前記記憶手段に記憶された複数のレジスタを特定する情報を参照して、前記複数のレジスタを復帰させるコードを生成するための第 2 のコード生成手段を含むアセンブラ。

【請求項 10】 前記第 1 のコード生成手段は、前記コード読込手段によって読み込まれたコードが第 1 のマクロ命令である場合、前記コードに含まれる複数のレジスタのうち、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタ以外のレジスタを退避するコードを生成する、請求項 9 記載のアセンブラ。

【請求項 11】 ソースプログラムからコードを読み込むステップと、

前記コードが第 1 のマクロ命令である場合、前記コードに含まれる複数のレジスタを特定するための情報を記憶し、前記複数のレジスタを退避するコードを生成するステップと、

前記読み込まれたコードが第 2 のマクロ命令である場合、前記記憶された複数のレジスタを特定する情報を参照して、前記複数のレジスタを復帰させるコード

を生成するステップを含むアセンブリ方法。

【請求項 1 2】 前記複数のレジスタを退避するコードを生成するステップは、前記読み込まれたコードが第 1 のマクロ命令である場合、前記読み込まれたコードに含まれる複数のレジスタのうち、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタ以外のレジスタを退避するコードを生成するステップを含む、請求項 1 1 記載のアセンブリ方法。

【請求項 1 3】 コンピュータにアセンブリ方法を実行させるプログラムが記録されたコンピュータで読取可能な記録媒体であって、

前記アセンブリ方法は、ソースプログラムからコードを読み込むステップと、

前記コードが第 1 のマクロ命令である場合、前記コードに含まれる複数のレジスタを特定するための情報を記憶し、前記複数のレジスタを退避するコードを生成するステップと、

前記読み込まれたコードが第 2 のマクロ命令である場合、前記記憶された複数のレジスタを特定する情報を参照して、前記複数のレジスタを復帰させるコードを生成するステップを含む、アセンブリプログラムを記録した記録媒体。

【請求項 1 4】 前記複数のレジスタを退避するコードを生成するステップは、前記読み込まれたコードが第 1 のマクロ命令である場合、前記読み込まれたコードに含まれる複数のレジスタのうち、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタ以外のレジスタを退避するコードを生成するステップを含む、請求項 1 3 記載のアセンブリプログラムを記録した記録媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、マイクロプロセッサおよびプログラムをそのマイクロプロセッサが実行可能な機械語に変換するアセンブラに関し、特に、スタックに対してデータを効率的に退避／復帰するマイクロプロセッサ並びにアセンブラ、その方法およびそのプログラムを記録した記録媒体に関する。

【0 0 0 2】

【従来の技術】

近年、パーソナルコンピュータ等の情報処理装置を始めとして、様々な電子機器にマイクロプロセッサが使用されている。一般に、マイクロプロセッサがプログラムを実行する場合、レジスタに格納された現在の値を一時的に退避するために、メモリの一部をスタック領域に割り当てる。このスタックに対するデータの退避／復帰は、L I F O (Last In First Out) 方式によって行なわれる。

【0003】

このようなマイクロプロセッサにおいては、最後にスタックに保存されたデータの位置を管理するためのレジスタとして、スタックポインタが使用される。このスタックポインタの実装方法として、専用のレジスタとしてマイクロプロセッサに実装される場合と、汎用レジスタのうち1本がスタックポインタとして使用される場合とが挙げられる。命令長が16ビット以上のマイクロプロセッサにおいては、レジスタの内容をスタックに退避する操作、またはスタックに格納されたデータをレジスタへ復帰させる操作をマイクロプロセッサに行なわせるために、ソフトウェアによって対象となるレジスタとその操作とを指定するのが一般的である。

【0004】

【発明が解決しようとする課題】

近年主流となりつつあるR I S C (Reduced Instruction Set Computer) 方式のマイクロプロセッサにおいて、マイクロプロセッサを構成するレジスタは、メモリに対して直接データの読み出し／書き込みが行なえるレジスタ（以下、データ用レジスタと呼ぶ。）と、メモリに対して直接データの読み出し／書き込みが行なえないレジスタ（以下、総称して制御レジスタと呼ぶ。）とに大きく分類される。

【0005】

制御レジスタの内容をスタックに退避したり、スタックに格納されたデータを制御レジスタに復帰させたりする場合、制御レジスタからワークレジスタへ一旦データを転送した後にスタックに書き込むか、スタックに格納されたデータをワークレジスタに一旦読み出した後に制御レジスタへ転送する必要がある。したがって、データ用レジスタの内容をスタックに退避したり、スタックに格納された

データをデータ用レジスタに復帰させる場合と比較して、プログラムのステップ数が余分に必要になるという問題点があった。

【0006】

一方、全てのレジスタがメモリに対して直接データの読み出し／書き込みが行なえるタイプのマイクロプロセッサにおいては、レジスタの内容をスタックに退避したり、スタックに格納されたデータをレジスタに復帰させたりする際のプログラムのステップ数は、上述したRISC方式のマイクロプロセッサと比較して少なくなる。しかし、レジスタを選択する回路構成が複雑になるため、RISC方式のマイクロプロセッサと比較して、動作周波数を高くすることが難しくなるという問題点があった。

【0007】

本発明は、上記問題点を解決するためになされたものであり、第1の目的は、RISC方式のマイクロプロセッサに採用される回路構成を採用しつつ、複数の制御レジスタの退避／復帰を短いステップ数のプログラムで実現できるマイクロプロセッサを提供することである。

【0008】

第2の目的は、簡単なマクロ命令で複雑なスタック操作を行なえるアセンブラ、その方法およびそのプログラムを記録した記録媒体を提供することである。

【0009】

第3の目的は、複数の制御レジスタの退避／復帰の整合性を自動的に管理することが可能なアセンブラ、その方法およびそのプログラムを記録した記録媒体を提供することである。

【0010】

【課題を解決するための手段】

請求項1に記載のマイクロプロセッサは、命令コードのフェッチを制御するための制御手段と、フェッチされた命令コードをデコードするためのデコード手段と、デコード手段によるデコード結果に基づいて、メモリのアドレスを演算するためのアドレス演算手段と、デコード手段によるデコード結果に基づいて、データを演算するためのデータ演算手段とを含み、データ演算手段は、制御手段によ

ってフェッチされた1つのオペレーションコードを有する1つの命令コードに対応して、レジスタ間のデータ転送と、レジスタとメモリとの間のデータ転送とを実行する。

【0011】

1つの命令コードでレジスタ間のデータ転送と、レジスタとメモリとの間のデータ転送とを実行できるため、所定の処理を行なうプログラムのステップ数を削減することが可能となる。

【0012】

請求項2に記載のマイクロプロセッサは、請求項1記載のマイクロプロセッサであって、データ演算手段は、制御手段によってフェッチされた1つの退避命令に対応して、第1のレジスタに格納されたデータをメモリに転送し、第2のレジスタに格納されたデータを第1のレジスタに転送する。

【0013】

したがって、直接メモリにデータを退避できない第2のレジスタに格納されたデータの退避を、1命令分のスループットで実行することが可能となる。

【0014】

請求項3に記載のマイクロプロセッサは、請求項2記載のマイクロプロセッサであって、データ演算手段は、第2のレジスタに格納されたデータを第1のレジスタに転送した後、スタックポインタの値をデクリメントする。

【0015】

第2のレジスタに格納されたデータを第1のレジスタに転送した後、スタックポインタの値がデクリメントされるため、スタック操作をさらに容易に行なうことが可能となる。

【0016】

請求項4に記載のマイクロプロセッサは、請求項1記載のマイクロプロセッサであって、データ演算手段は、制御手段によってフェッチされた1つの復帰命令に対応して、第1のレジスタに格納されたデータを第2のレジスタに転送し、メモリに格納されたデータを前記第1のレジスタに転送する。

【0017】

したがって、直接メモリからデータを復帰できない第2のレジスタへのデータの復帰を、1命令分のスループットで実行することが可能となる。

【0018】

請求項5に記載のマイクロプロセッサは、請求項4に記載のマイクロプロセッサであって、データ演算手段は、メモリに格納されたデータを第1のレジスタに転送した後、スタックポインタの値をインクリメントする。

【0019】

メモリに格納されたデータを第1のレジスタに転送した後、スタックポインタの値がインクリメントされるため、スタック操作をさらに容易に行なうことが可能となる。

【0020】

請求項6に記載のマイクロプロセッサは、請求項2～5のいずれかに記載のマイクロプロセッサであって、第1のレジスタは、データ演算手段に実装されるワークレジスタである。

【0021】

したがって、第2のレジスタとメモリとの間のデータ転送を、ワークレジスタを媒介として行なうことが可能となる。

【0022】

請求項7に記載のマイクロプロセッサは、請求項2～6のいずれかに記載のマイクロプロセッサであって、第2のレジスタは、アドレス演算手段または制御手段に実装される制御レジスタである。

【0023】

したがって、第1のレジスタを媒介として制御レジスタとメモリとの間のデータ転送を行なうことが可能となる。また、第1のレジスタをワークレジスタとしてデータ演算手段に実装し、第2のレジスタを制御レジスタとしてアドレス演算手段に実装することによって、RISC方式のマイクロプロセッサの回路構成を採用することが可能となる。

【0024】

請求項8に記載のマイクロプロセッサは、請求項1～7のいずれかに記載のマ

マイクロプロセッサであって、データ演算手段は、制御手段によってフェッチされた1つの退避命令に対して、第1のレジスタに格納されたデータをメモリに転送し、スタックポインタの値をそのまま保持するための手段を含む。

【0025】

第1のレジスタに格納されたデータをメモリに転送し、スタックポインタの値をそのまま保持する退避命令を実行可能とすることにより、スタックポインタが最後に退避されたデータを指し示すようにすることが可能となる。

【0026】

請求項9に記載のアセンブラは、ソースプログラムからコードを読み込むためのコード読込手段と、複数のレジスタを特定するための情報を記憶するための記憶手段と、コード読込手段によって読み込まれたコードが第1のマクロ命令である場合、コードに含まれる複数のレジスタを特定するための情報を記憶手段に記憶し、複数のレジスタを退避するコードを生成するための第1のコード生成手段と、コード読込手段によって読み込まれたコードが第2のマクロ命令である場合、記憶手段に記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するための第2のコード生成手段とを含む。

【0027】

第1のコード生成手段は、第1のマクロ命令から複数のレジスタを退避するコードを生成するので、1つのマクロ命令で複雑なスタック操作を扱うことが可能となる。また、第2のコード生成手段は、記憶手段に記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するので、1つのマクロ命令で複雑なスタック操作を扱うことができ、複数のレジスタの退避／復帰の整合性を自動的に管理することが可能となる。

【0028】

請求項10に記載のアセンブラは、請求項9に記載のアセンブラであって、第1のコード生成手段は、コード読込手段によって読み込まれたコードが第1のマクロ命令である場合、コードに含まれる複数のレジスタのうち、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタ以外のレジスタを退避するコードを生成する。

【 0 0 2 9 】

したがって、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタが自動的にスタックに退避される場合に、冗長なコードが生成されるのを防止することが可能となる。

【 0 0 3 0 】

請求項 1 1 に記載のアセンブリ方法は、ソースプログラムからコードを読み込むステップと、コードが第 1 のマクロ命令である場合、コードに含まれる複数のレジスタを特定するための情報を記憶し、複数のレジスタを退避するコードを生成するステップと、読み込まれたコードが第 2 のマクロ命令である場合、記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するステップとを含む。

【 0 0 3 1 】

第 1 のマクロ命令から複数のレジスタを退避するコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことが可能となる。また、読み込まれたコードが第 2 のマクロ命令である場合、記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことができ、複数のレジスタの退避／復帰の整合性を自動的に管理することが可能となる。

【 0 0 3 2 】

請求項 1 2 に記載のアセンブリ方法は、請求項 1 1 に記載のアセンブリ方法であって、複数のレジスタを退避するコードを生成するステップは、読み込まれたコードが第 1 のマクロ命令である場合、読み込まれたコードに含まれる複数のレジスタのうち、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタ以外のレジスタを退避するコードを生成するステップを含む。

【 0 0 3 3 】

したがって、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタが自動的にスタックに退避される場合に、冗長なコードが生成されるのを防止することが可能となる。

【 0 0 3 4 】

請求項 1 3 に記載の記録媒体は、コンピュータにアセンブリ方法を実行させるプログラムが記録されたコンピュータで読取可能な記録媒体であって、アセンブリ方法は、ソースプログラムからコードを読み込むステップと、コードが第 1 のマクロ命令である場合、コードに含まれる複数のレジスタを特定するための情報を記憶し、複数のレジスタを退避するコードを生成するステップと、読み込まれたコードが第 2 のマクロ命令である場合、記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するステップとを含む。

【 0 0 3 5 】

第 1 のマクロ命令から複数のレジスタを退避するコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことが可能となる。また、読み込まれたコードが第 2 のマクロ命令である場合、記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことができ、複数のレジスタの退避／復帰の整合性を自動的に管理することが可能となる。

【 0 0 3 6 】

請求項 1 4 に記載の記録媒体は、請求項 1 3 に記載のアセンブリプログラムが記録された記録媒体であって、複数のレジスタを退避するコードを生成するステップは、読み込まれたコードが第 1 のマクロ命令である場合、読み込まれたコードに含まれる複数のレジスタのうち、レジスタ間のデータ転送の媒介として使用されるレジスタ以外のレジスタを退避するコードを生成するステップを含む。

【 0 0 3 7 】

したがって、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタが自動的にスタックに退避される場合に、冗長なコードが生成されるのを防止することが可能となる。

【 0 0 3 8 】

【発明の実施の形態】

（実施の形態 1）

図 1 は、本発明の実施の形態 1 におけるマイクロプロセッサを構成するレジス

タセットを説明するための図である。本実施の形態においては、マイクロプロセッサのデータ長が16ビットの場合について説明するが、これに限られるものではない。

【0039】

図1(a)に示すレジスタR0～R3は、4本の演算ソースレジスタであり、演算のソースが格納される。レジスタTR0～TR3は、4本のワークレジスタであり、アドレスやデータ（演算ソースや演算結果）が一時的に保存されるレジスタである。A0およびA1は、それぞれ40ビットのアキュムレータである。アキュムレータA0およびA1はそれぞれ、演算ソースまたは演算結果の上位16ビットが保持されるA0HおよびA1Hと、演算ソースまたは演算結果の下位16ビットが保持されるA0LおよびA1Lと、上位ビットからあふれたビットが保持される8ビット長のガードビットA0GおよびA1Gとを含む。

【0040】

図1(b)に示すレジスタAR0～AR3は、4本のアドレスレジスタであり、メモリアクセスの際のアドレスが格納される。レジスタAMD0～AMD3は、4本のアドレッシングモードレジスタであり、それぞれアドレスレジスタAR0～AR3を用いたアドレッシングモードが格納される。レジスタAR_SELは、AR割当レジスタであり、アドレスレジスタの選択に使用される。

【0041】

レジスタMOD_SおよびMOD_Eは、それぞれモジュロアドレッシングのための制御レジスタである。MOD_Sはモジュロスタートアドレスを保持し、MOD_Eはモジュロエンドアドレスを保持する。SPはスタックポインタであり、スタックの先頭アドレスを保持する。レジスタAR_PAGEは、ページ指定レジスタであり、メモリをページ単位で指定する場合のページ先頭アドレスを格納する。

【0042】

PCはプログラムカウンタであり、マイクロプロセッサが現在実行しているプログラムのアドレスを保持する。PSWはプロセッサ状態語であり、マイクロプロセッサを制御するためのフラグ等を格納する。BPCおよびBPSWは、それ

それバックアップ用のプログラムカウンタおよびプロセッサ状態語であり、割り込み等の事象が発生した場合にPCおよびPSWの値がそれぞれ自動的にコピーされる。

【0043】

DPCおよびDPSWは、それぞれデバッグ用のプログラムカウンタおよびプロセッサ状態語であり、デバッグ用割り込み事象が発生した場合にPCおよびPSWの値が自動的にコピーされる。レジスタPCLINKはリンクレジスタであり、サブルーチンからの戻りアドレスを保持する。LP_CTおよびREP_CTはそれぞれループカウンタおよびリピートカウンタであり、ブロックリピートの回数および単一命令リピートの回数をそれぞれ保持する。

【0044】

レジスタLP_SおよびLP_Eは、それぞれブロックリピートの先頭アドレスおよび終了アドレスを指定するレジスタである。レジスタPC_BRKは、ハードウェアブレークポイントを指定する際に利用されるレジスタである。レジスタINT_Sは、割り込みステータスレジスタである。レジスタCR00～CR63は、周辺I/O（Input/Output）との間のデータの入出力に利用されるI/Oマップドレジスタである。なお、レジスタINT_SおよびCR00～CR63は、外部に接続された入出力デバイスを制御する際に使用されるレジスタであり、本実施の形態に直接関連するものではないため、詳細な説明は行なわない。なお、以下の説明においては、1つの命令コードに1つのオペレーションコードが含まれるものとする。

【0045】

図2は、本実施の形態におけるマイクロプロセッサの概略構成を示すブロック図である。このマイクロプロセッサは、命令メモリ43からフェッチされた命令をデコードする命令デコード部39と、命令メモリ43に格納された命令のフェッチを制御するPCU（Program Control Unit）部40と、Xメモリ44またはYメモリ45にアクセスする際のアドレスを演算するAAU（Address Arithmetic Unit）部41と、データの演算を行なうDAU（Data Arithmetic Unit）部42とを含む。命令メモリ43は、命令のバイナリコードを格納する。また、X

メモリ 4 4 および Y メモリ 4 5 は、被演算値、演算結果等のデータを格納する。

【 0 0 4 6 】

命令デコード部 3 9 は、命令メモリ 4 3 からフェッチされた命令コードをデコードし、命令コードに応じた制御信号 P 4 6、A 4 7 および D 4 8 をそれぞれ P C U 部 4 0、A A U 部 4 1 および D A U 部 4 2 へ出力する。P C U 部 4 0 は、命令デコード部 3 9 から出力された制御信号 P 4 6 に基づいて、次にフェッチすべき命令が格納されるアドレスを、アドレスバス 5 6 を介して命令メモリ 4 3 へ出力する。A A U 部 4 1 は、命令デコード部 3 9 から出力された制御信号 A 4 7 に基づいて、必要に応じて読み出すべきデータが格納されるアドレスを生成し、アドレスバス 5 7 を介して X メモリ 4 4 および Y メモリ 4 5 へ出力する。

【 0 0 4 7 】

D A U 部 4 2 は、1 7 ビット×1 7 ビットの乗算を行なう乗算器 4 9、4 0 ビットの 2 つのデータに演算を行なう A L U (Arithmetic and Logic Unit) 5 0 および 4 0 ビットの入力データに対して左右に 1 6 ビットのシフト演算を行なうシフタ 5 1 を含む。D A U 部 4 2 は、命令デコード部 3 9 から出力された制御信号 D 4 8 に基づいて、上述したレジスタに保持された値や、データバス 5 3 を介して X メモリ 4 4 または Y メモリ 4 5 から読み出した値に対して、乗算、加減算またはシフト演算等を行なう。

【 0 0 4 8 】

図 1 に示すレジスタセットは、図 2 に示す P C U 部 4 0、A A U 部 4 1 および D A U 部 4 2 のいずれかに実装されている。P C U 部 4 0 は、1 3 個のレジスタ P C、P S W、B P C、B P S W、D P C、D P S W、P C L I N K、L P _ C T、R E P _ C T、L P _ S、L P _ E、P C _ B R K および I N T _ S を有するレジスタ群 6 0 を含む。

【 0 0 4 9 】

A A U 部 4 1 は、1 3 個のレジスタ A R 0 ~ A R 3、A M D 0 ~ A M D 3、A R _ S E L、M O D _ S、M O D _ E、S P および A R _ P A G E を有するレジスタ群 6 1 を含む。また、D A U 部 4 2 は、4 個のレジスタ T R 0 ~ T R 3 を有するレジスタ群 6 2 と、4 個のレジスタ R 0 ~ T R 3 を有するレジスタ群 6 3 と

、2個のレジスタA0およびA1を有するレジスタ群64とを含む。

【0050】

DAU部42は、レジスタ群63の各レジスタからデータを受け、そのデータをMPY49の一方の入力、ALU50の一方の入力またはSFT51の一方の入力へ転送するためのデータバスD1と、レジスタ群63の各レジスタからデータを受け、そのデータをMPY49の他方の入力、ALU50の他方の入力またはSFT51の他方の入力へ転送するためのデータバスD2と、MPX49、ALU50またはSFT51から出力されたデータを受け、そのデータをレジスタ群64の各レジスタへ転送するためのデータバスD3とを含む。DAU部42はさらに、データバスD6を含み、このデータバスD6を介してレジスタ群64の各レジスタからALU50、SFT51のいずれかへのデータ転送を行なう。

【0051】

また、DAU部42はさらにデータバスD4を含み、このデータバスD4を介してレジスタ群63の各レジスタへ入力されるデータの転送、レジスタ群64の各レジスタから出力されるデータの転送、およびレジスタ群62の各レジスタに入出力されるデータの双方向の転送が行なわれる。さらには、このデータバスD4およびデータバス53を介して、DAU部42とXメモリ44またはYメモリ45との間の双方向のデータ転送も行なわれる。

【0052】

マイクロプロセッサはさらに、レジスタ群60～64間の双方向のデータ転送を行なうためのデータバスD5を含む。算術演算、論理演算、シフト等の演算命令を実行する場合には、レジスタ群63または64内の選択されたレジスタからデータバスD1および（または）D2またはD6へデータが供給され、その演算結果がデータバスD3を介してレジスタ群64内の所定のレジスタへ格納される。

【0053】

レジスタ間のデータ転送命令を実行する場合には、データバスD5を介してレジスタ間のデータ転送が行なわれる。特に、レジスタ群64内のレジスタA0またはA1からレジスタ群63内のレジスタへのデータ転送のときには、バスD4

が使用される。

【0054】

ロード命令の場合には、Xメモリ44またはYメモリ45のデータが、データバスD4を介してレジスタ群62またはレジスタ群63へ転送される。ストア命令の場合には、レジスタ群62またはレジスタ群64のデータが、データバスD4および53を介してXメモリ44へ転送される。

【0055】

次に、上述した本実施の形態におけるマイクロプロセッサが命令メモリ43に格納されたプログラムを実行する手順について説明する。まず、PCU部40はフェッチすべき命令コードが格納されたアドレスを、アドレスバス56を介して命令メモリ43へ出力する。命令デコード部39は、命令メモリ43から出力された命令コードを、データバス52を介して読み出して命令コードのデコードを行なう。命令デコード部39は、命令コードのデコード結果に基づいて制御信号P46、A47およびD48を出力する。

【0056】

PCU部40は、制御信号P46に基づいて次にフェッチすべき命令が格納されるアドレスを生成し、アドレスバス56を介して命令メモリ43へ出力する。AAU部41は、制御信号A47がXメモリ44およびYメモリ45の一方または両方へのアクセスを示すものであれば、読み出しまたは書き込みのためのアドレスを生成し、アドレスバス57を介してXメモリ44およびYメモリ45へ出力する。

【0057】

DAU部42は、制御信号D48に基づいて演算処理を行なう。たとえば、制御信号D48がXメモリ44またはYメモリ45からデータを読み出して演算を行なうものであれば、DAU部42はXメモリ44またはYメモリ45から出力されたデータを読み込んで、そのデータに対して演算処理を行なう。また、制御信号D48がレジスタの内容を演算し、演算結果をXメモリ44へ書き込むものであれば、DAU部42は演算結果をデータバス53および54を介してメモリ44に出力する。

【 0 0 5 8 】

図 4 は、本実施の形態におけるマイクロプロセッサによって処理される演算命令の一例を示す図である。また、図 5 は、本実施の形態におけるマイクロプロセッサによって処理される転送命令、シーケンス制御命令および特殊命令の一例を示す図である。これらの命令の内容は、それぞれ命令の右側に説明されているので、詳細な説明は行なわない。

【 0 0 5 9 】

図 4 および図 5 に示す命令のうち、LOAD 命令およびSTORE 命令は、アドレスレジスタAR0～AR3を用いてアドレッシングを行ない、メモリとレジスタとの間のデータ転送を行なう命令である。このように、命令中にアドレスレジスタを指定するようにして、メモリのアクセスの自由度を高くしている。ただし、DAU部42内に実装されたレジスタの一部のみが指定可能である。そのため、これらの命令を用いてDAU部42以外のブロックに実装されたレジスタの内容を退避するためには、レジスタ間転送を行なうmv命令を用いて一旦DAU部42に実装されたレジスタに転送した後、レジスタとメモリとの間のデータ転送を行なう必要がある。図 6 は、LOAD 命令およびSTORE 命令において指定可能なレジスタの一覧を示す図である。

【 0 0 6 0 】

図 7 は、STORE 命令を用いたレジスタ(TR0, AR0)の退避動作を行なうプログラムの一例を示す図である。図 7 の(1)および(2)は、アドレスレジスタAR3およびそれに対応するアドレッシングモードレジスタAMD3を初期化している。すなわち、アドレスレジスタAR3に#STACK_BOTTOMを代入し、アドレッシングモードレジスタAMD3に#DEC_1を代入している。なお、#STACK_BOTTOMはレジスタ退避領域の最上位アドレスを表わす定数であり、#DEC_1はAR3の値が読み出される毎に1だけデクリメントされることを指定する定数を表わしている。

【 0 0 6 1 】

図 7 の(3)は、ワークレジスタTR0の値が、Xメモリ44のアドレスレジスタAR3によって示されるアドレスに格納されることを示している。この命令

が実行されると、DAU部42に実装されているワークレジスタTR0の値が、データバス53および54を介してDAU部42からXメモリ44へ出力される。また、AAU部41に実装されているアドレスレジスタAR3の値が、アドレスバス57を介してAAU部41からXメモリ44へ出力される。そして、AAU部41においてアドレスレジスタAR3の値がデクリメントされる。

【0062】

図7の(4)は、アドレスレジスタAR0の値が、ワークレジスタTR0に転送されることを示している。この命令が実行されると、AAU部41に実装されているアドレスレジスタAR0の値が、DAU部42に実装されているワークレジスタTR0に転送される。また、図7の(5)は(3)と同様に、ワークレジスタTR0の値が、Xメモリ44のアドレスレジスタAR3によって示されるアドレスに格納されることを示している。

【0063】

図8は、図7に示すプログラムが実行されたときのスタックの動作を示す図である。図7の(1)および(2)に示す命令コードが実行されると、アドレスレジスタAR3に格納されるアドレスが図8(a)に示す位置を示すようになる。そして、図7の(3)に示す命令コードが実行されると、アドレスレジスタAR3によって示されるアドレスにワークレジスタTR0の値が格納され、アドレスレジスタAR3の値がデクリメントされる。

【0064】

図7の(4)に示す命令コードが実行されると、アドレスレジスタAR0の値がワークレジスタTR0に転送される(図8(c)参照)。そして、図7の(5)に示す命令コードが実行されると、図8(d)に示すように、アドレスレジスタAR3によって示されるアドレスにワークレジスタTR0(AR0)の値が格納され、アドレスレジスタAR3の値がデクリメントされる。なお、LOAD命令を用いたレジスタへの復帰動作は、図7および図8に示す動作と逆の動作となるため、詳細な説明は行なわない。

【0065】

図7および図8に示す転送方式において、制御レジスタの内容をスタックに退

避するためには、制御レジスタの内容をワークレジスタを介して転送する必要があるため、制御レジスタN個の内容を転送するためには(2×N)ステップ分の命令コードが必要となる。

【0066】

本実施の形態におけるマイクロプロセッサにおいては、図1に示したレジスタセット以外に、次に説明するPOP命令、PUSH命令およびPUT命令を実装する。図9は、このPOP命令、PUSH命令およびPUT命令のニーモニックおよびその動作を説明するための図である。PUSH命令およびPOP命令は、任意のレジスタを指定することができる。また、PUT命令は、レジスタを指定することができない。

【0067】

図9(a)に示すように、POP命令においてレジスタ指定がない場合(1)には、Xメモリ44のスタックポインタで指定されたアドレスに格納されるデータが、DAU部42に実装されたワークレジスタTR0へ転送され、スタックポインタの値がインクリメントされる。また、POP命令においてレジスタ指定がある場合(2)には、ワークレジスタTR0の値がPOP命令によって指定されるレジスタに転送され、Xメモリ44のスタックポインタで指定されたアドレスに格納されるデータが、データバス53および54を介してDAU部42に実装されたワークレジスタTR0に転送された後、スタックポインタの値がインクリメントされる。

【0068】

図9(b)に示すように、PUSH命令においてレジスタ指定がない場合(1)には、スタックポインタの値がデクリメントされる。また、PUSH命令においてレジスタ指定がある場合(2)には、DAU部42に実装されたワークレジスタTR0に格納されるデータが、Xメモリ44のスタックポインタで指定されたアドレスに格納され、PUSH命令によって指定されたレジスタの値がワークレジスタTR0に転送された後、スタックポインタの値がデクリメントされる。

【0069】

図9(c)に示すように、PUT命令が実行されると、DAU部42に実装さ

れたワークレジスタT R 0に格納されるデータが、Xメモリ44のスタックポインタで指定されたアドレスに転送される。なお、P U T命令が実行されてもスタックポインタの値は更新されない。

【0070】

なお、上述した説明において、P O P命令のときにスタックポインタの値がインクリメントされ、P U S H命令のときにスタックポインタの値がデクリメントされる場合を説明したが、逆にP O P命令のときにスタックポインタの値がデクリメントされ、P U S H命令のときにスタックポインタの値がインクリメントされるようにしても良い。

【0071】

上述したP O P命令、P U S H命令、P U T命令の詳細動作を以下に説明する。図3は、本実施の形態におけるマイクロプロセッサのパイプライン処理を説明するための図である。これらの命令は、動作クロックの1サイクルずつ、それぞれ命令フェッチI F、命令デコードDおよび命令実行Eの3段のパイプラインで処理される。

【0072】

オペランドありP O P命令は、時刻（A）に示す動作クロックの立上リエッジでトリガされて、レジスタ群62内のレジスタT R 0に記憶されたデータがデータバスD5に出力され、同時に、Xメモリ44内のS Pで示された領域に記憶されたデータがデータバス53経由でD4に出力される。また、時刻（B）に示す動作クロックの立上リエッジでトリガされて、データバスD5に出力されたT R 0のデータが、レジスタ群60、61、63および64のうちP O P命令で指定されたレジスタに書込まれ、同時に、データバスD4に出力されたデータがT R 0に書込まれ、さらにS Pの値がインクリメントされる。これらのデータ転送は、時刻（C）に示す動作クロックの立下がりエッジでされても良い。

【0073】

オペランドなしP O P命令は、時刻（A）に示す動作クロックの立上リエッジでトリガされて、Xメモリ44内のS Pで示された領域に記憶されたデータが、データバス53経由でD4に出力される。また、時刻（B）に示す動作クロック

のエッジでトリガされて、データバスD 4 の値はT R 0 に書込まれ、同時に、S P の値はインクリメントされる。

【0 0 7 4】

オペランドありP U S H命令は、時刻（A）に示す動作クロックの立上リエッジでトリガされて、レジスタT R 0 に記憶されたデータが、データバスD 4 経由でデータバス5 3 に出力されXメモリ4 4 内のS P で示された領域に書込まれる。同時に、レジスタ群6 0、6 1、6 3 および6 4 のうちP U S H命令で指定されたレジスタに記憶されたデータが、データバスD 5 に出力される。また、時刻（B）に示す動作クロックの立上リエッジでトリガされて、データバスD 5 に出力されたデータがT R 0 に書込まれ、同時に、S P の値はデクリメントされる。

【0 0 7 5】

オペランドなしP U S H命令は、時刻（B）に示す動作クロックの立上リエッジでトリガされて、S P の値がインクリメントされる。

【0 0 7 6】

P U T命令は、時刻（A）に示す動作クロックの立上リエッジでトリガされて、レジスタT R 0 に記憶されたデータがデータバス5 3 およびD 4 を介してXメモリ4 4 内のS P で示された領域へ書込まれる。

【0 0 7 7】

図1 0 は、上述したP O P命令、P U S H命令およびP U T命令を使用したプログラムの一例を示す図である。このプログラムは、演算ソースレジスタR 0 およびアドレスレジスタA R 0 を退避した後、復帰するものである。図1 0 の（1）に示す“p u s h”は、スタックポインタの値をデクリメントして空き領域を指すことを示している。図1 0 の（2）に示す“p u s h R 0”は、ワークレジスタT R 0 の値が、Xメモリ4 4 のスタックポインタで指定されたアドレスに格納され、演算ソースレジスタR 0 の値がワークレジスタT R 0 に転送された後、スタックポインタの値がデクリメントされることを示している。

【0 0 7 8】

図1 0 の（3）に示す“p u s h A R 0”は、ワークレジスタT R 0 の値が、Xメモリ4 4 のスタックポインタで指定されたアドレスに格納され、アドレス

レジスタAR0の値がワークレジスタTR0に転送された後、スタックポインタの値がデクリメントされることを示している。図10の(4)に示す“put”は、ワークレジスタTR0の値が、Xメモリ44のスタックポインタで指定されたアドレスに格納されることを示している。

【0079】

図10の(5)に示す“pop”は、Xメモリ44のスタックポインタで示されたアドレスに格納されるデータが、ワークレジスタTR0に転送された後、スタックポインタの値がインクリメントされることを示している。図10の(6)に示す“pop AR0”は、ワークレジスタTR0の値がアドレスレジスタAR0に転送され、Xメモリ44のスタックポインタで示されたアドレスに格納されるデータが、ワークレジスタTR0に転送された後、スタックポインタの値がインクリメントされることを示している。また、図10の(7)に示す“pop

R0”は、ワークレジスタTR0の値が演算ソースレジスタR0に転送され、Xメモリ44のスタックポインタで示されたアドレスに格納されるデータが、ワークレジスタTR0に転送された後、スタックポインタの値がインクリメントされることを示している。

【0080】

図11は、図10に示すプログラムを実行したときのスタックの動作を説明するための図である。図10に示すプログラムを実行する前は、図11(a)に示す位置をスタックポインタが指し示している。図10の(1)に示す命令を実行すると、スタックポインタの値がデクリメントされて、図11(b)に示すようにスタックポインタが空き領域を指し示すようになる。次に、図10の(2)に示す命令が実行されると、ワークレジスタTR0の値がXメモリ44のスタックポインタで示されたアドレスに格納される。そして、図11(c)に示すように演算ソースレジスタR0の値がワークレジスタTR0に転送された後、スタックポインタの値がデクリメントされる。

【0081】

次に、図10の(3)に示す命令が実行されると、ワークレジスタTR0の値(R0)がXメモリ44のスタックポインタで示されたアドレスに格納される。

そして、図 1 1 (d) に示すようにアドレスレジスタ A R 0 の値がワークレジスタ T R 0 に転送された後、スタックポインタの値がデクリメントされる。次に、図 1 0 の (4) に示す命令が実行されると、ワークレジスタ T R 0 の値 (A R 0) が X メモリ 4 4 のスタックポインタで示されたアドレスに格納される。なお、このときスタックポインタの値は更新されない (図 1 1 (e) 参照)。

【 0 0 8 2 】

次に、図 1 0 の (5) に示す命令が実行されると、X メモリ 4 4 のスタックポインタで示されたアドレスに格納されるデータ (A R 0) がワークレジスタ T R 0 に転送される。そして、図 1 1 (f) に示すようにスタックポインタの値がインクリメントされる。次に、図 1 0 の (6) に示す命令が実行されると、ワークレジスタ T R 0 の値がアドレスレジスタ A R 0 に転送され、X メモリ 4 4 のスタックポインタで示されたアドレスに格納されるデータ (R 0) がワークレジスタ T R 0 に転送される。そして、図 1 1 (g) に示すようにスタックポインタの値がインクリメントされる。

【 0 0 8 3 】

最後に、図 1 0 の (7) に示す命令が実行されると、ワークレジスタ T R 0 の値が演算ソースレジスタ R 0 に転送され、X メモリ 4 4 のスタックポインタで示されたアドレスに格納されるデータがワークレジスタ T R 0 に転送される。そして、図 1 1 (h) に示すようにスタックポインタの値がインクリメントされる。このように、P U S H 命令、P O P 命令および P U T 命令を使用してスタック操作を行なうことにより、N 個の制御レジスタの退避は (N + 2) ステップで実現でき、N 個の制御レジスタの復帰は (N + 1) ステップで実現できる。

【 0 0 8 4 】

以上説明したように、本実施の形態におけるマイクロプロセッサによれば、1 つの退避命令 (P U S H 命令) で制御レジスタからワークレジスタへのデータ転送と、ワークレジスタから X メモリ 4 4 へのデータ転送とが行なわれるようにし、1 つの復帰命令 (P O P 命令) で X メモリ 4 4 からワークレジスタへのデータ転送と、ワークレジスタから制御レジスタへのデータ転送とが行なわれるようにしたので、短いステップ数で複数の制御レジスタの退避および復帰が行なわれる

ようになった。また、退避命令および復帰命令によって上述した動作が行なわれるため、A A U 部 4 1 に直接データバスを接続する必要がなくなり、R I S C 方式のプロセッサの回路構成を採用することができ、マイクロプロセッサ内部の回路構成を簡略化することが可能となる。

【 0 0 8 5 】

(実施の形態 2)

本発明の実施の形態 2 は、実施の形態 1 において説明したマイクロプロセッサが実行可能な機械語にプログラムを変換するアセンブラに関する。このアセンブラは、パーソナルコンピュータやワークステーション等のコンピュータにアセンブリプログラムを実行させることによって実現される。

【 0 0 8 6 】

図 1 2 は、アセンブラを実現するコンピュータの構成例を示すブロック図である。このコンピュータは、コンピュータ本体 1、グラフィックディスプレイ装置 2、FD (Floppy Disk) 4 が装着される FD ドライブ 3、キーボード 5、マウス 6、CD-ROM (Compact Disc-Read Only Memory) 8 が装着される CD-ROM 装置 7、およびネットワーク通信装置 9 を含む。

【 0 0 8 7 】

アセンブリプログラムは、FD 4 または CD-ROM 8 等の記憶媒体によって供給される。アセンブリプログラムはコンピュータ本体 1 によって実行され、プログラムによって作成されたプログラムを実施の形態 1 において説明したマイクロプロセッサが実行可能な機械語に変換する。また、アセンブリプログラムは他のコンピュータより通信回線を経由し、コンピュータ本体 1 に供給されてもよい。

【 0 0 8 8 】

また、コンピュータ本体 1 は、CPU 1 0、ROM (Read Only Memory) 1 1、RAM (Random Access Memory) 1 2 およびハードディスク 1 3 を含む。CPU 1 0 は、グラフィックディスプレイ装置 2、磁気テープ装置 3、キーボード 5、マウス 6、CD-ROM 装置 7、ネットワーク通信装置 9、ROM 1 1、RAM 1 2 またはハードディスク 1 3 との間でデータを入出力しながら処理を行なう。

。FD4またはCD-ROM8に記録されたアセンブリプログラムは、CPU10によりFDドライブ3またはCD-ROM装置7を介して一旦ハードディスク13に格納される。CPU10は、ハードディスク13から適宜アセンブリプログラムをRAM12にロードして実行することによって処理を行なう。

【0089】

図13は、本実施の形態におけるアセンブラによって処理されるマクロ命令を説明するための図である。図13(a)の(1)に示すマクロ命令“MPUSH R0, AR0;”は、演算ソースレジスタR0およびアドレスレジスタAR0を退避することを示している。なお、“MPUSH”以降に退避すべきレジスタが記述されるものとし、レジスタの数は特に限定がないものとする。

【0090】

図13(a)の(2)に示すマクロ命令“MPOP”は、直前に記述されたマクロ命令“MPUSH”に対応するマクロ命令であり、マクロ命令“MPUSH”によって退避されたレジスタの内容を全て復帰させるものである。なお、マクロ命令の記述方法はこれらに限られるものではなく、マクロ命令が展開された後の命令コードが均等であれば同一と考えられるべきである。

【0091】

図13(b)は、図13(a)に示すマクロ命令“MPUSH”を展開したときの命令コードを示している。また、図13(c)は、図13(a)に示すマクロ命令“MPOP”を展開したときの命令コードを示している。図13(b)および図13(c)に示すプログラムの内容は、図10に示すプログラムの内容と同じであるので、詳細な説明は繰り返さない。

【0092】

図14は、本実施の形態におけるアセンブラの機能的構成の概略を示すブロック図である。このアセンブラは、ソースファイルからコードを1行ずつ読み込み、読み込んだコードの解釈等を行なうコード解釈部20と、MPUSH命令を命令コードに展開するMPUSH命令展開部21と、MPOP命令を命令コードに展開するMPOP命令展開部22と、MPUSH命令およびMPOP命令以外の命令のコード生成を行なうコード生成部23とを含む。

【 0 0 9 3 】

図 1 5 は、本実施の形態におけるアセンブラの処理手順を示している。まず、コード解釈部 2 0 は、ソースファイルからコードを 1 行読み込み、そのソース行のコードにエラーがあるか否か、およびそのソース行が最終行であるか否かを判定する (S 1)。そのソース行のコードにエラーがあるか、またはそのソース行が最終行であれば (S 1, N G)、コード解釈部 2 0 は処理を終了する。

【 0 0 9 4 】

また、そのソース行が最終行でなく、コードにエラーがなければ (S 1, O K)、ソース解釈部 2 0 はそのコードが M P U S H 命令であるか否かを判定する (S 2)。そのコードが M P U S H 命令であれば (S 2, Y e s)、M P U S H 命令展開部 2 1 はその M P U S H 命令を展開し (S 3)、展開結果を R A M 1 2 に格納する。その後、ステップ S 1 へ戻って以降の処理を繰り返す。

【 0 0 9 5 】

また、コードが M P U S H 命令でなければ (S 2, N o)、そのコードが M P O P 命令であるか否かを判定する (S 4)。そのコードが M P O P 命令であれば (S 4, Y e s)、M P O P 命令展開部 2 2 はその M P O P 命令を展開し (S 5)、ステップ S 1 へ戻って以降の処理を繰り返す。また、そのコードが M P O P 命令でなければ (S 4, N o)、コード生成部 2 3 は通常のコード生成を行ない (S 6)、そのコードを R A M 1 2 に格納する。その後、ステップ S 1 へ戻って以降の処理を繰り返す。

【 0 0 9 6 】

図 1 6 は、図 1 5 のステップ S 3 (M P U S H 命令の展開) の処理をさらに詳細に説明するためのフローチャートである。まず、M P U S H 命令展開部 2 1 は、オペランドなし (レジスタ指定がない) P U S H 命令を生成し、そのコードを R A M 1 2 に格納する (S 3 1)。次に、M P U S H 展開部 2 1 は、M P U S H 命令のオペランドをチェックする (S 3 2)。M P U S H 命令展開部 2 1 は、M P U S H 命令に記述されたオペランドを順にチェックし、未処理のオペランドがあれば (S 3 2, O K)、そのオペランドを含んだ P U S H 命令を生成してそのコードを R A M 1 2 に格納するとともに、そのオペランドを L I F O 2 4 に格納

する（S 3 3）。そして、ステップ S 3 2 へ戻って以降の処理を繰り返す。また、未処理のオペランドがなければ（S 3 2, NG）、PUT 命令を生成してそのコードを RAM 1 2 に格納する（S 3 4）。

【0 0 9 7】

このようにして、MPUSH 命令が展開されて、そのコードが RAM 1 2 に格納される。たとえば、図 1 3（a）の（1）に示す MPUSH 命令の場合であれば、ステップ S 3 3 においてまずオペランド R 0 が抽出されて、“push R 0” のコードが生成される。次に、オペランド A R 0 が抽出されて、“push A R 0” のコードが生成される。なお、図 1 6 に示す LIFO 2 4 は、図 1 2 に示す RAM 1 2 またはハードディスク 1 3 に形成される。

【0 0 9 8】

図 1 7 は、図 1 5 のステップ S 5（MPOP 命令の展開）の処理をさらに詳細に説明するためのフローチャートである。まず、MPOP 命令展開部 2 2 は、オペランドなし（レジスタ指定がない）POP 命令を生成し、そのコードを RAM 1 2 に格納する（S 5 1）。次に、MPOP 展開部 2 2 は、MPUSH 命令の展開時に LIFO 2 4 に格納されたオペランドを読み出し（S 5 2）、未処理のオペランド（レジスタ）の有無を判定する（S 5 3）。

【0 0 9 9】

MPOP 命令展開部 2 2 は、未処理のオペランドがあれば（S 5 3, OK）、そのオペランドを含んだ POP 命令を生成してそのコードを RAM 1 2 に格納する（S 5 4）。そして、ステップ S 5 2 へ戻って以降の処理を繰り返す。また、未処理のオペランドがなければ（S 5 3, NG）、処理を終了する。なお、図 1 7 に示す LIFO 2 4 は、図 1 2 に示す RAM 1 2 または CPU 1 0 内のレジスタ群に相当する。

【0 1 0 0】

このようにして、MPOP 命令が展開されて、そのコードが RAM 1 2 に格納される。たとえば、図 1 3（a）の（2）に示す MPOP 命令の場合であれば、LIFO 2 4 に“R 0”および“A R 0”が格納されているので、ステップ S 5 4 においてまずオペランド A R 0 が抽出されて、“pop A R 0”のコードが

生成される。次に、オペランド R 0 が抽出されて、“p o p R 0”のコードが生成される。

【0 1 0 1】

以上説明したように、本実施の形態におけるアセンブラによれば、マクロ命令を実施の形態 1 において説明したマイクロプロセッサが実行可能なコードに展開するようにしたので、退避および復帰すべきレジスタを含んだマクロ命令を記述するだけで一連のスタック操作を行なうコードを生成することが可能となる。したがって、プログラマはスタック操作の整合性等の確認を行なう必要がなくなり、ソフトウェア開発における生産性を向上させることが可能となった。

【0 1 0 2】

（実施の形態 3）

上述した実施の形態 2 のアセンブラにおいては、ワークレジスタ T R 0 がレジスタ転送の媒介として使用される。図 1 1 の説明から分かるように、ワークレジスタ T R 0 の退避の有無に関わらず、ワークレジスタ T R 0 の値が自動的にスタックに格納される。本実施の形態におけるアセンブラは、この特徴を利用したものである。

【0 1 0 3】

本実施の形態におけるアセンブラの機能的構成は、図 1 4 に示す実施の形態 2 におけるアセンブラの機能的構成と比較して M P U S H 命令展開部の機能のみが異なる。また、本実施の形態におけるアセンブラの処理手順は、図 1 5 に示す実施の形態 2 におけるアセンブラの処理手順と同じである。したがって、重複する構成および機能の詳細な説明は繰り返さない。なお、本実施の形態における M P U S H 命令展開部の参照符号を 2 1' として説明する。

【0 1 0 4】

図 1 8 は、本実施の形態におけるアセンブラによって処理されるマクロ命令を説明するための図である。図 1 8 (a) の (1) に示すマクロ命令 “M P U S H T R 0 , A R 0 ;” は、ワークレジスタ T R 0 およびアドレスレジスタ A R 0 を退避することを示している。なお、“M P U S H” 以降に退避すべきレジスタが記述されるものとし、レジスタの数は特に限定がないものとする。

【0105】

図18(a)の(2)に示すマクロ命令“MPOP”は、直前に記述されたマクロ命令“MPUSH”に対応するマクロ命令であり、マクロ命令“MPUSH”によって退避されたレジスタの内容を全て復帰させるものである。なお、マクロ命令の記述方法はこれらに限られるものでなく、マクロ命令が展開された後の命令コードが均等であれば同一と考えられるべきである。

【0106】

図18(b)は、図18(a)に示すマクロ命令“MPUSH”を展開したときの命令コードを示している。また、図18(c)は、図18(a)に示すマクロ命令“MPOP”を展開したときの命令コードを示している。ワークレジスタTR0はスタックに自動的に格納されることになるため、ワークレジスタTR0に対応した退避命令は生成されない。また、ワークレジスタTR0以外のレジスタを復帰させると、自動的にワークレジスタTR0も復帰されるため、ワークレジスタTR0に対応した復帰命令も生成されない。

【0107】

図19は、図15のステップS3(MPUSH命令の展開)の処理をさらに詳細に説明するためのフローチャートである。まず、MPUSH命令展開部22'は、オペランドなし(レジスタ指定がない)PUSH命令を生成し、そのコードをRAM12に格納する(S61)。次に、MPUSH展開部21'は、MPUSH命令のオペランドをチェックする(S62)。

【0108】

MPUSH命令展開部21'は、MPUSH命令に記述されたオペランドを順にチェックし、未処理のオペランドがあれば(S62, OK)、そのオペランドがワークレジスタTR0であるか否かを判定する(S63)。そのオペランドがワークレジスタTR0であれば(S63, Yes)、そのままステップS62へ戻って以降の処理を繰り返す。

【0109】

また、オペランドがワークレジスタTR0でなければ(S63, No)、そのオペランドを含んだPUSH命令を生成してそのコードをRAM12に格納する

とともに、そのオペランドをLIFO24に格納する(S64)。そして、ステップS62へ戻って以降の処理を繰り返す。また、未処理のオペランドがなければ(S62, NG)、PUT命令を生成してそのコードをRAM12に格納し(S65)、処理を終了する。

【0110】

以上説明したように、本実施の形態におけるアセンブラによれば、レジスタとメモリとの間のデータ転送において媒介として使用されるレジスタを退避／復帰する命令を生成しないようにしたので、レジスタの退避／復帰における冗長なコードが生成されるのを防止することができ、プログラムのステップ数の削減、処理速度の向上および使用メモリサイズの削減を図ることが可能となった。

【0111】

今回開示された実施の形態は、すべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【0112】

【発明の効果】

請求項1に記載のマイクロプロセッサによれば、1つの命令コードでレジスタ間のデータ転送と、レジスタとメモリとの間のデータ転送とを実行できるため、所定の処理を行なうプログラムのステップ数を削減することが可能となった。

【0113】

請求項2に記載のマイクロプロセッサによれば、直接メモリにデータを退避できない第2のレジスタに格納されたデータの退避を、1命令分のスループットで実行することが可能となった。

【0114】

請求項3に記載のマイクロプロセッサによれば、第2のレジスタに格納されたデータを第1のレジスタに転送した後、スタックポインタの値がデクリメントされるため、スタック操作をさらに容易に行なうことが可能となった。

【0115】

請求項4に記載のマイクロプロセッサによれば、直接メモリからデータを復帰できない第2のレジスタへのデータの復帰を、1命令分のスループットで実行することが可能となった。

【0116】

請求項5に記載のマイクロプロセッサによれば、メモリに格納されたデータを第1のレジスタに転送した後、スタックポインタの値がインクリメントされるため、スタック操作をさらに容易に行なうことが可能となった。

【0117】

請求項6に記載のマイクロプロセッサによれば、第2のレジスタとメモリとの間のデータ転送を、ワークレジスタを媒介として行なうことが可能となった。

【0118】

請求項7に記載のマイクロプロセッサによれば、第1のレジスタを媒介として制御レジスタとメモリとの間のデータ転送を行なうことが可能となった。また、第1のレジスタをワークレジスタとしてデータ演算手段に実装し、第2のレジスタを制御レジスタとしてアドレス演算手段または制御手段に実装することによって、RISC方式のマイクロプロセッサの回路構成を採用することが可能となった。

【0119】

請求項8に記載のマイクロプロセッサによれば、第1のレジスタに格納されたデータをメモリに転送し、スタックポインタの値をそのまま保持する退避命令を実行可能とすることにより、スタックポインタが最後に退避されたデータを指し示すようにすることが可能となった。

【0120】

請求項9に記載のアセンブラによれば、第1のコード生成手段は、第1のマクロ命令から複数のレジスタを退避するコードを生成するので、1つのマクロ命令で複雑なスタック操作を扱うことが可能となった。また、第2のコード生成手段は、記憶手段に記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するので、1つのマクロ命令で複雑なスタック操作を扱うことができ、複数のレジスタの退避／復帰の整合性を自動的に管理す

ることが可能となった。

【 0 1 2 1 】

請求項 1 0 に記載のアセンブラによれば、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタが自動的にスタックに退避される場合に、冗長なコードが生成されるのを防止することが可能となった。

【 0 1 2 2 】

請求項 1 1 に記載のアセンブリ方法によれば、第 1 のマクロ命令から複数のレジスタを退避するコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことが可能となった。また、読み込まれたコードが第 2 のマクロ命令である場合、記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことができ、複数のレジスタの退避／復帰の整合性を自動的に管理することが可能となった。

【 0 1 2 3 】

請求項 1 2 に記載のアセンブリ方法によれば、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタが自動的にスタックに退避される場合に、冗長なコードが生成されるのを防止することが可能となった。

【 0 1 2 4 】

請求項 1 3 に記載の記録媒体によれば、第 1 のマクロ命令から複数のレジスタを退避するコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことが可能となった。また、読み込まれたコードが第 2 のマクロ命令である場合、記憶された複数のレジスタを特定する情報を参照して、複数のレジスタを復帰させるコードを生成するので、1 つのマクロ命令で複雑なスタック操作を扱うことができ、複数のレジスタの退避／復帰の整合性を自動的に管理することが可能となった。

【 0 1 2 5 】

請求項 1 4 に記載の記録媒体によれば、レジスタとメモリとの間のデータ転送の媒介として使用されるレジスタが自動的にスタックに退避される場合に、冗長なコードが生成されるのを防止することが可能となった。

【図面の簡単な説明】

【図 1】 本発明の実施の形態 1 におけるマイクロプロセッサを構成するレジスタセットを説明するための図である。

【図 2】 本発明の実施の形態 1 におけるマイクロプロセッサの概略構成を示すブロック図である。

【図 3】 本発明の実施の形態 1 におけるマイクロプロセッサのパイプライン処理を説明するための図である。

【図 4】 本発明の実施の形態 1 におけるマイクロプロセッサによって処理される演算命令の一例を示す図である。

【図 5】 本発明の実施の形態 1 におけるマイクロプロセッサによって処理される転送命令、シーケンス制御命令および特殊命令の一例を示す図である。

【図 6】 LOAD 命令および STORE 命令において指定可能なレジスタの一覧を示す図である。

【図 7】 STORE 命令を用いたレジスタの退避動作を行なうプログラムの一例を示す図である。

【図 8】 図 7 に示すプログラムが実行されたときのスタックの動作を示す図である。

【図 9】 POP 命令、PUSH 命令および PUT 命令のニーモニックおよびその動作を説明するための図である。

【図 10】 POP 命令、PUSH 命令および PUT 命令を使用したプログラムの一例を示す図である。

【図 11】 図 10 に示すプログラムを実行したときのスタックの動作を説明するための図である。

【図 12】 本発明の実施の形態 2 におけるアセンブラを実現するコンピュータの構成例を示すブロック図である。

【図 13】 本発明の実施の形態 2 におけるアセンブラによって処理されるマクロ命令を説明するための図である。

【図 14】 本発明の実施の形態 2 におけるアセンブラの概略構成を示すブロック図である。

【図 1 5】 本発明の実施の形態 2 におけるアセンブラの処理手順を説明するためのフローチャートである。

【図 1 6】 図 1 5 のステップ S 3 の処理をさらに詳細に説明するためのフローチャートである。

【図 1 7】 図 1 5 のステップ S 5 の処理をさらに詳細に説明するためのフローチャートである。

【図 1 8】 本発明の実施の形態 3 におけるアセンブラによって処理されるマクロ命令を説明するための図である。

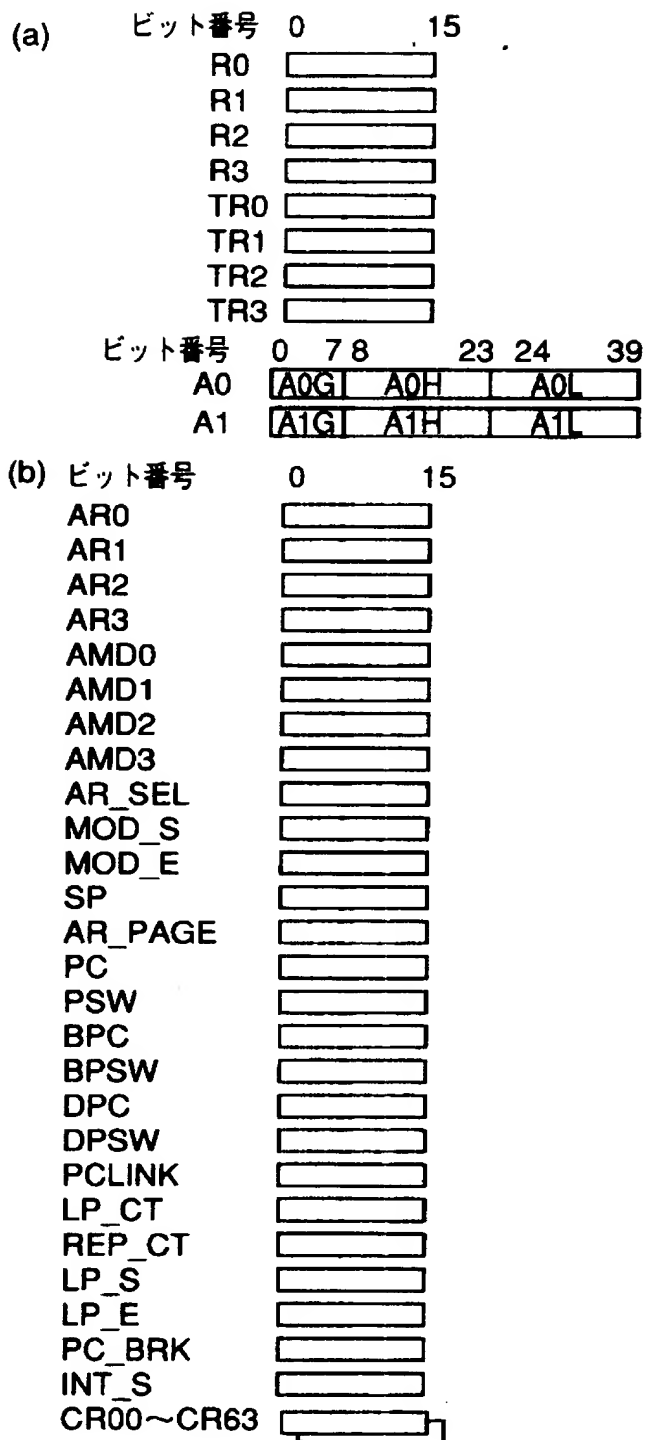
【図 1 9】 図 1 5 のステップ S 3 の処理をさらに詳細に説明するためのフローチャートである。

【符号の説明】

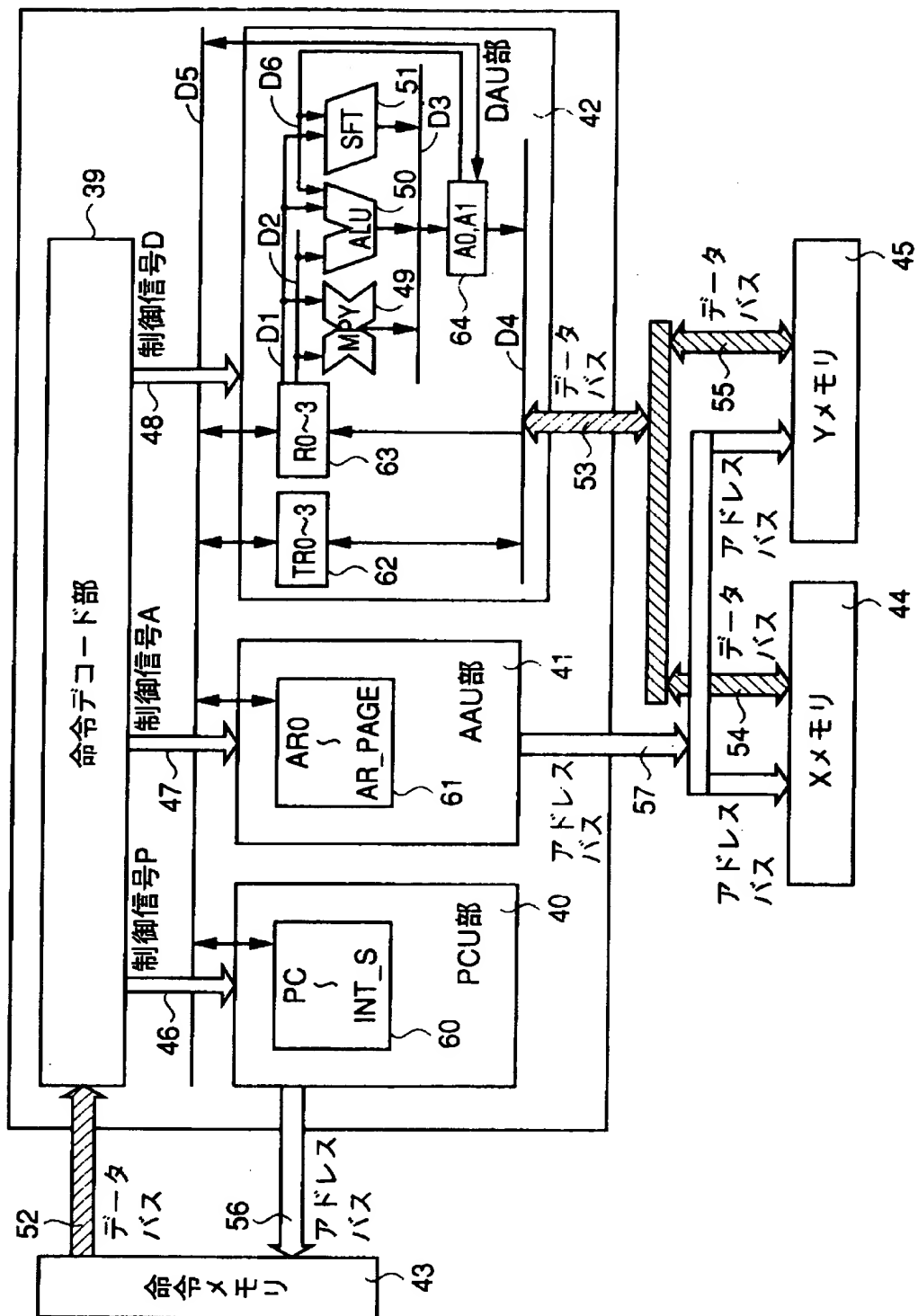
1 コンピュータ本体、2 グラフィックディスプレイ装置、3 FDドライブ、4 FD、5 キーボード、6 マウス、7 CD-ROM装置、8 CD-ROM、9 ネットワーク通信装置、10 CPU、11 ROM、12 RAM、13 ハードディスク、20 コード解釈部、21 MPUSH命令展開部、22 MPOP命令展開部、23 コード生成部、24 LIFO、39 命令デコード部、40 PCU部、41 AAU部、42 DAU部、43 命令メモリ、44 Xメモリ、45 Yメモリ、49 乗算器、50 ALU、51 シフタ。

【書類名】 図面

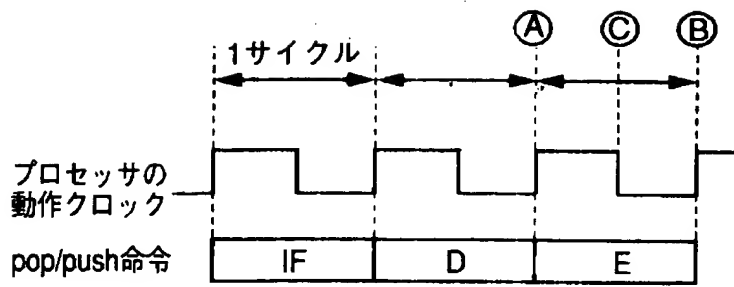
【図 1】



【図 2】



【図 3】



【図 4】

演算命令

mul	Multiply
muluu	Multiply unsigned operands
mac	Multiply and add
macuu	Multiply unsigned operands and add
macsu	Multiply signed operand by unsigned operand and add
macsul	Multiply signed operand by unsigned operand and add with shift right
macsuh	Multiply signed operand by unsigned operand and add with shift left
msub	Multiply and sub
msubuu	Multiply unsigned operands and sub
msubsul	Multiply signed operand by unsigned operand and add with shift right
msubsuh	Multiply signed operand by unsigned operand and add with shift left
add	Add a register to acc high
addl	Add a register to acc low
sub	Subtract a register from acc high
subl	Subtract a register from acc low
min	Set minimum value of acch or reg to accumulator
max	Set maximum value of accl or reg to accumulator
amin	Set minimum value to dest-acc
amax	Set maximum value to dest-acc
sra	Shift arithmetic right or left an accumulator
srl	Shift logical right or left an accumulator
and	And
or	Or
xor	Xor
nop	No operation
trfh	Transfer to an accumulator high
trfl	Transfer to an accumulator low
trf	Transfer to an accumulator
aadd	Add accumulators
asub	Subtract src-acc from dest-acc
sadd	Add dest-acc and src-acc with shift
abs	Absolute an accumulator
neg	Negate an accumulator
test	Test an accumulator(acc<0:set Nflag, acc==0:set Zflag)
md	Round an accumulator
not	Not an accumulator

【図 5】

転送命令

mv Copy one word from a register to a register
 ldi Load immediate
 ld Load
 st Store
 push Push to stack
 put Put to stack
 pop Pop from stack

シーケンス制御命令

jmp Jump
 call Jump & link
 loopi Set loop counter and start hardware DO loop
 loop Start hardware DO loop
 repeati Set repeat counter and repeat next instruction
 repeat Repeat next instruction
 return Return from subroutine
 reit Return from EIT
 rtd Return from debugger EIT

特殊命令

adr_set Set AR_SEL register
 mvin Move from IO registers
 mvout Move to IO registers
 slave Transit to slav mode
 noop No operation

【図 6】

	指定可能なレジスタ
LOAD命令	R0, R1, R2, R3, TR0, TR1, TR2, TR3
STORE命令	TR0, TR1, TR2, TR3, A0H, A0L, A1H, A1L

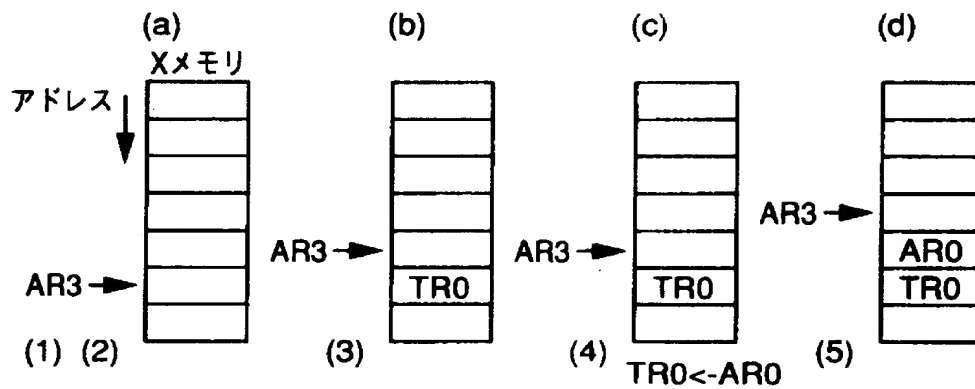
【図 7】

```

LDI AR3, #STACK_BOTTOM ; (1)
LDI AMD3, #DEC_1        ; (2)
. . . . .
ST TR0, X:AR3            ; (3)
MV TR0, AR0              ; (4)
ST TR0, X:AR3            ; (5)

```

【図 8】



【図 9】

- (a) POP
 [mnemonics]
 (1) pop
 (2) pop ra
 [operation]
 (1) tr0 = x_memory[sp];
 sp++;
 (2) ra = tr0;
 tr0 = x_memory[sp];
 sp++;
- (b) PUSH
 [mnemonics]
 (1) push
 (2) push ra
 [operation]
 (1)
 sp--;
 (2) x_memory[sp] = tr0;
 tr0 = ra;
 sp--;
- (c) PUT
 [mnemonics]
 put
 [operation]
 x_memory[sp] = tr0;

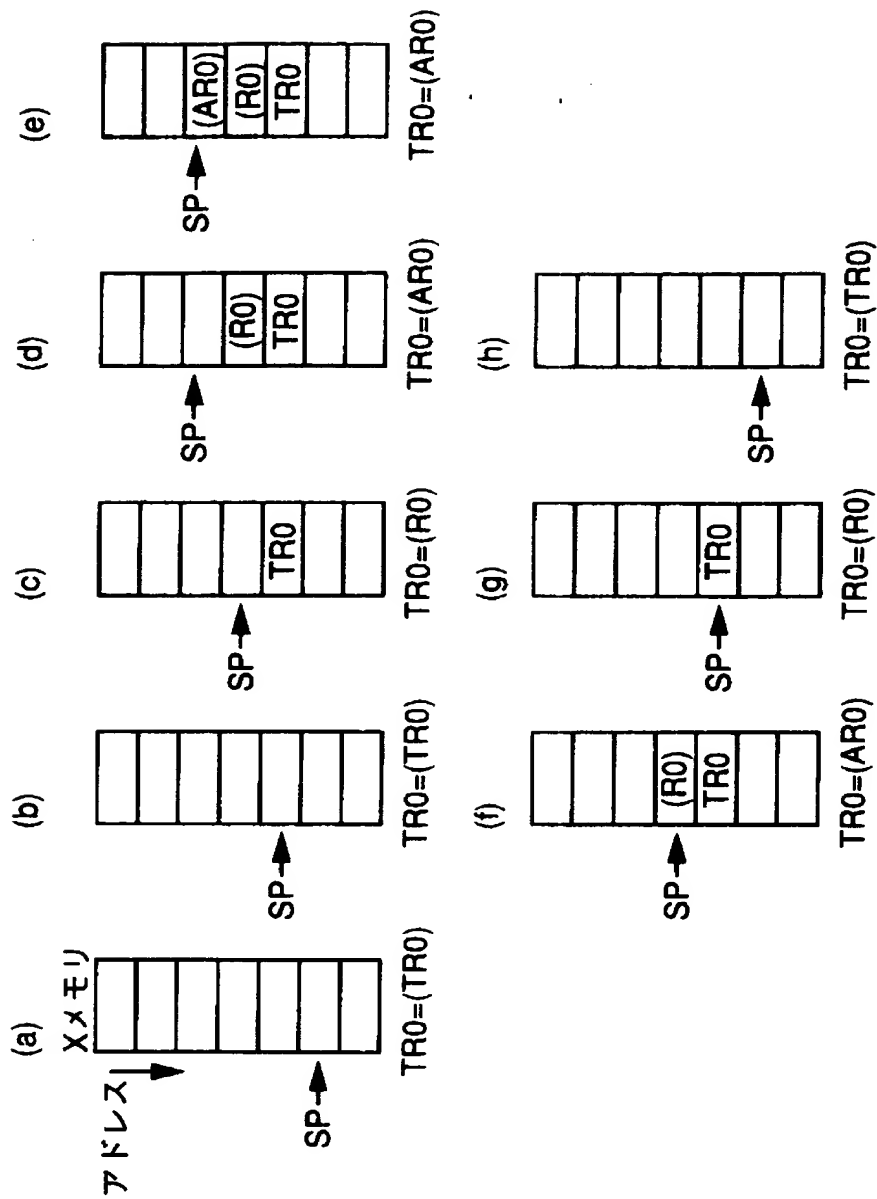
【図 10】

```

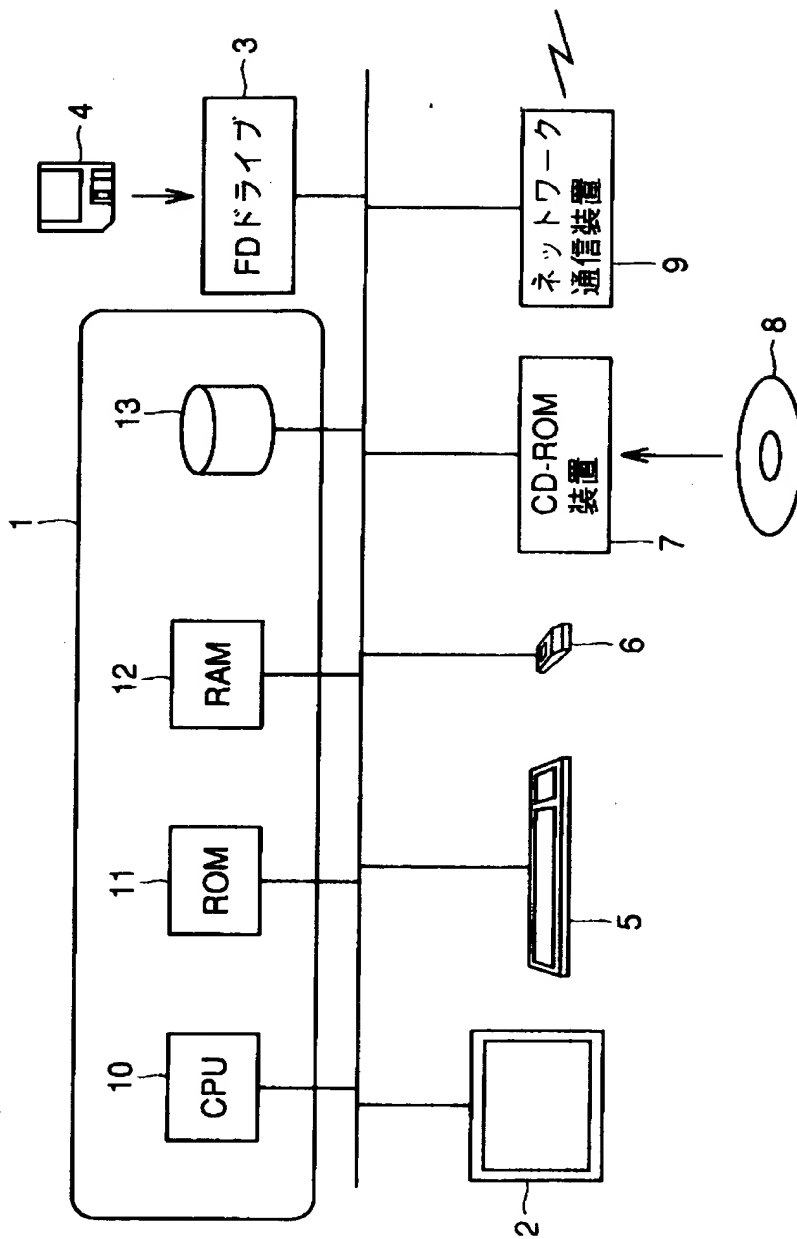
push          ; (1)
push R0       ; (2)
push AR0      ; (3)
put           ; (4)
. . . . .
pop           ; (5)
pop AR0       ; (6)
pop R0        ; (7)

```

【図 11】



【図 12】



【図 1 3】

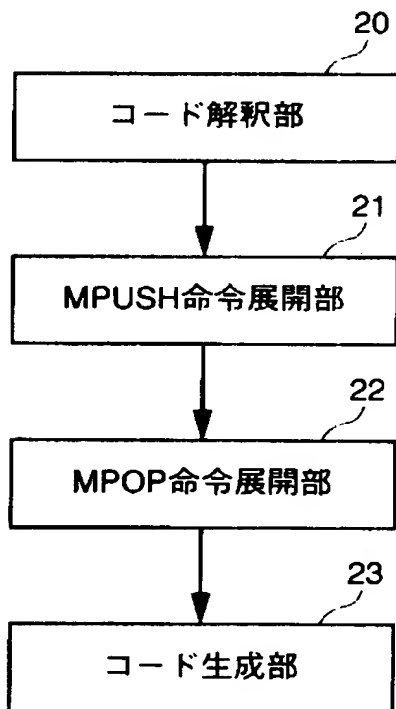
(a) MPUSH R0, AR0; (1)

 MPOP ;(2)

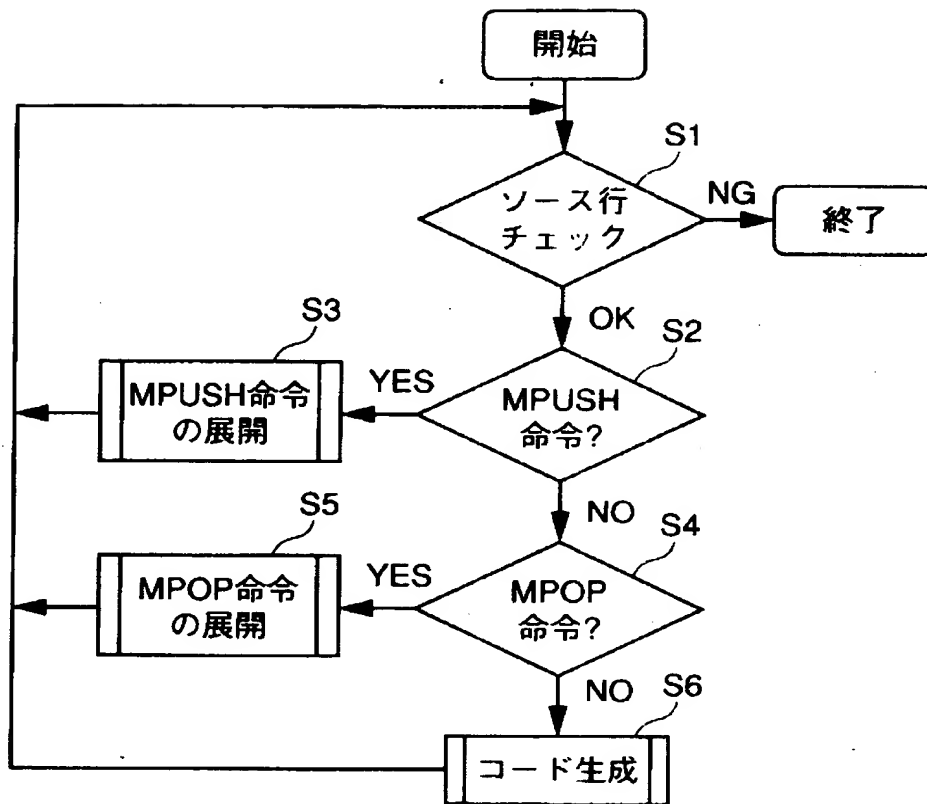
(b)
 push ;
 push R0 ;
 push AR0 ;
 put ;

(c)
 pop ;
 pop AR0 ;
 pop R0 ;

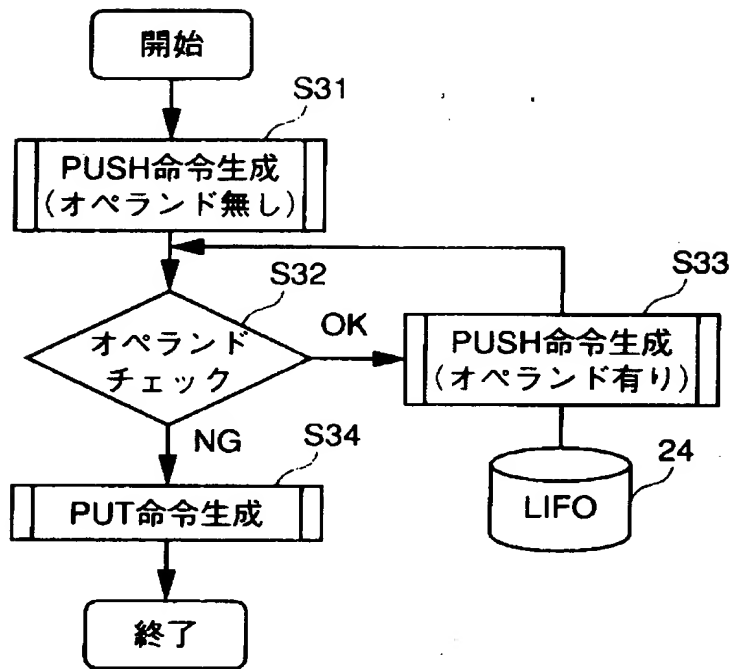
【図 1 4】



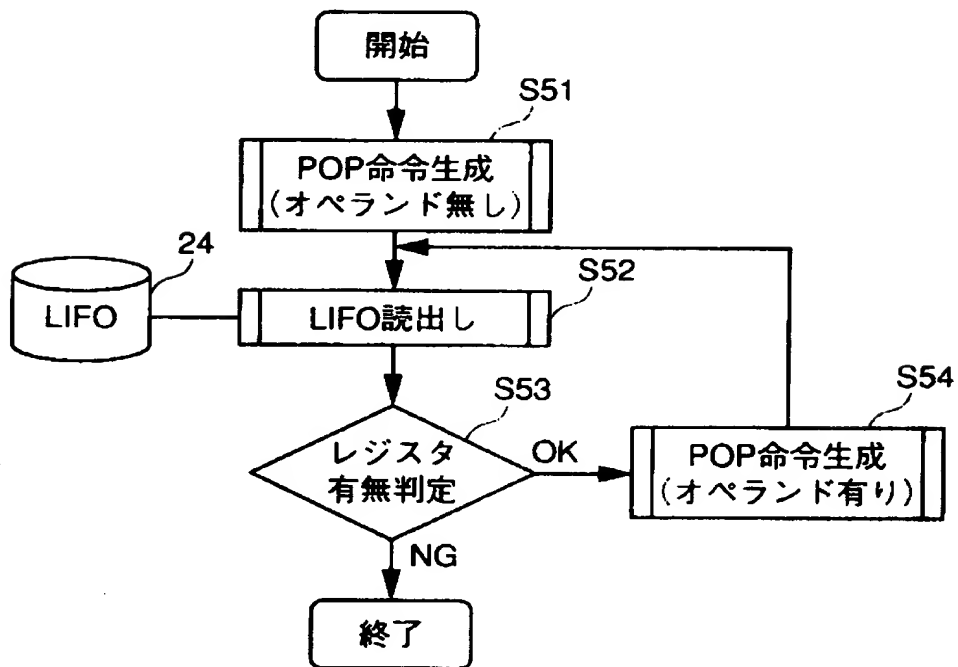
【図 15】



【図 16】



【図 17】



【図 1 8】

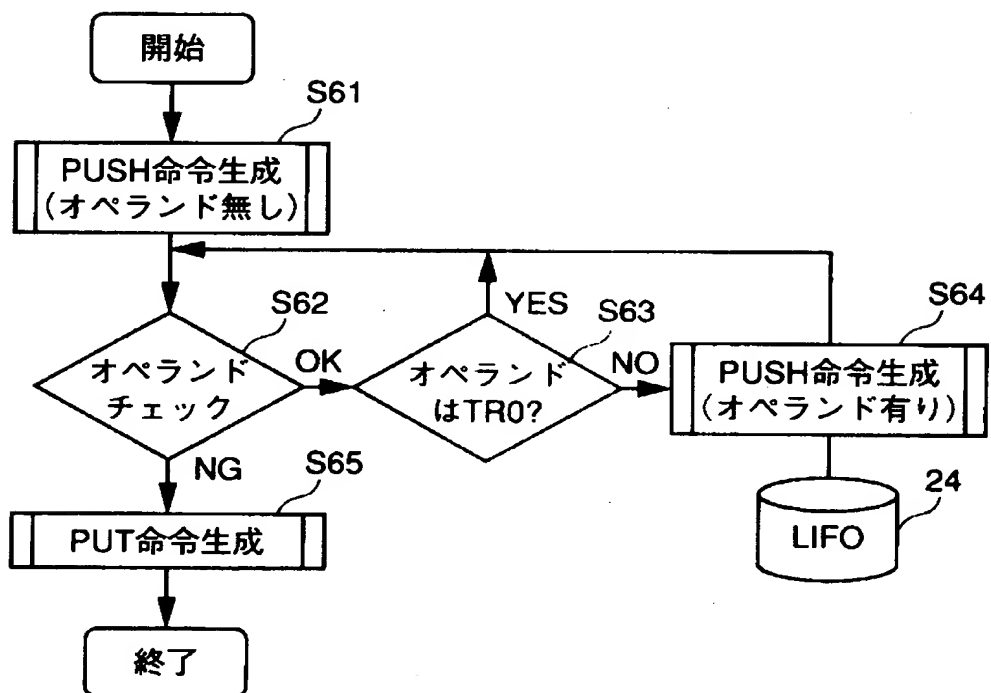
(a) MPUSH TR0, AR0; (1)

 MPOP ;(2)

(b) push ;
 push AR0 ;
 put ;

(c) pop ;
 pop AR0 ;

【図 1 9】



【書類名】 要約書

【要約】

【課題】 複数の制御レジスタの退避／復帰を短いステップ数のプログラムで実現できるマイクロプロセッサを提供すること。

【解決手段】 命令コードのフェッチを制御するPCU部40と、フェッチされた命令コードをデコードする命令デコード部39と、命令デコード部39によるデコード結果に基づいて、メモリのアドレスを演算するAAU部41と、1つの退避命令に対応して、制御レジスタとワークレジスタとの間のデータ転送と、ワークレジスタとXメモリ44との間のデータ転送とを行なうDAU部42とを含む。したがって、直接メモリにデータを退避できない制御レジスタに格納されたデータの退避を、1つの退避命令で実行することが可能となる。

【選択図】 図2

出 願 人 履 歴 情 報

識別番号 [000006013]

1. 変更年月日	1990年 8月24日
[変更理由]	新規登録
住 所	東京都千代田区丸の内2丁目2番3号
氏 名	三菱電機株式会社